

Examining the Impact of Platform Properties on Quality Attributes

Balwinder Sodhi and T.V. Prabhakar

Dept. of Computer Science and Engineering, IIT Kanpur, UP 208016 India
{sodhi, tvp}@cse.iitk.ac.in

Abstract—We examine and bring out the architecturally significant characteristics of various virtualization and cloud oriented platforms. The impact of such characteristics on the ability of guest applications to achieve various quality attributes (QA) has also been determined by examining existing body of architecture knowledge. We observe from our findings that efficiency, resource elasticity and security are among the most impacted QAs, and virtualization platforms exhibit the maximum impact on various QAs.

Index Terms—Software Architecture Design, Non-functional Requirements, Design Decisions, Quality Attributes, Cloud Computing, Virtualization, Design decision impact

I. INTRODUCTION

Virtualization and cloud oriented computing platforms have emerged as quite disruptive candidates for deployment and development of software applications. For designing the architecture of a complex software system, it is critical to understand important properties of target computing platforms. Equally important is to know the impact of these properties on various functional and non-functional aspects of an application. Towards that end, we have examined and analysed architectural aspects of virtualization and cloud based platforms.

For the said platforms and their underlying components, we determine what are the architecturally significant *characteristics* that are important from an application's perspective. We used the term *characteristics* here to collectively represent both functional features and the non-functional quality present in the platform. The impact of such characteristics on different QAs of a guest application has been determined by examining software architecture body of knowledge and performing benchmarking experiments. This impact knowledge is critical for assessing different platforms on a set of QA criteria.

A. Related work

Several researchers have explored different dimensions of virtualization and cloud based platforms. For instance, a dissection of the cloud into five main layers, and illustrating their interrelations and inter-dependency on constituent components has been discussed by Youseff et. al. [1]. The architectural requirements of cloud platforms have been discussed by Rimal et. al. [2]. These works examine different technical dimensions of cloud computing. However, from the standpoint of an application that wants to make exploit capabilities of such modern computing platforms, some questions still do not have clear answers. For example:

- 1) What are important characteristics of various platforms from the viewpoint of a guest application?
- 2) How do such characteristics impact various functional and non-functional aspects of guest application?

In the presented work we address the above questions.

This report is organized into four sections. In Section II we examine the characteristics of various computing platforms. We bring out and discuss impact of said characteristics on QAs in Section III. Report is concluded in Section IV.

II. COMPUTING PLATFORMS CHARACTERISTICS

A platform has characteristics which are typically determined and specified by

- Its functional attributes.
- Non-functional QAs that it assures.
- Design decisions or tactics employed in its architecture.

The presented work examines in detail the characteristics of two platforms viz. *Virtualization based* and *Cloud based*. In order to bring out such characteristics, following documentation artifacts were examined:

- Architecture description documents. They provide different views, e.g. design decision view, deployment view etc., for platform's architecture.
- Product specifications that describe functional and non-functional features.
- Benchmarking data if available.

The platforms and their important properties have been discussed in detail in sections below.

A. Virtualization Based

In this type of platforms the hardware resources are virtual. Typically, the computing environment is offered as Virtual Machine (VM) which the Virtual Machine Monitor (VMM) executes on a physical hardware shared with other VMs.

Virtualization can be of two kinds: a) OS based as in Solaris containers [3], and b) Virtual Machine Monitor (VMM) based as in Xen [4]. Further, the VMM based virtualization can be of two kinds: bare metal and hosted [5]. Certain VMMs also exist that take advantage of special purpose hardware features such as of recent Intel processors [6].

The internals of various types of virtualization platforms (such as [3]–[8]) were examined in detail to bring out their key properties. Following are the chief properties *common* across above said types of virtualization platforms:

- 1) Limited hardware control capabilities for VMs
- 2) Programmatic self-serviced provisioning
- 3) VM check-pointing and snap shots
- 4) Software abstraction of hardware
- 5) Multi-tenancy by logical partitioning of physical host into encapsulated software entities
- 6) Abstraction of hardware platform specific APIs and ABIs (e.g. VMware's products export an x86-based computer)
- 7) VM migration (both live and offline)

OS based virtualization platforms exhibit the following characteristics:

- 1) Guest OS (or kernel in some cases) in VM cannot be different from the host one.
- 2) Programs in VMs use the OSs normal system call interface (no emulation involved).
- 3) Tight integration with host OS, and less overhead in comparison to the VMMs
- 4) Isolation under a shared OS instance only
- 5) Privileged tasks only under the host OS control (e.g. loading a device driver, changing IP in a VM); else the application has to be modified to work in a VM.
- 6) Uses file-level copy-on-write mechanisms

Characteristics exhibited by VMM based virtualization platforms are as below:

- 1) No restrictions on guest OS in VM
- 2) Programs in VMs use VMM provided emulation instead of the OSs normal system call interface into underlying hardware
- 3) Allows full guest OS control in the VM
- 4) Uses block-level copy-on-write
- 5) Bare-metal VMM:
 - a) In some cases modification of guest OS needed for running in a VM
 - b) Thin and encapsulated hardware facing layer
- 6) Hosted VMM:
 - a) No modification needed to guest OS for running in a VM
 - b) VM runs a regular process inside host OS subject to host OS environment

B. Cloud Based

The functional features and available architecture and low-level implementation details of various cloud platforms were examined [9]–[16] to bring out main characteristics of the said platforms. The key characteristics of cloud based computing platforms (common across various cloud variants) are listed as below:

- 1) Programmatic provisioning of resources
- 2) Allows self-service provisioning
- 3) Shared underlying computing infrastructure via multi-tenancy
- 4) Lack of standards for key services such as security, VM control and management among others.
- 5) Computing as a utility accessible over the network
- 6) Geographic location transparent to clients

- 7) Political/legal jurisdiction transparent to clients
- 8) Measured service
- 9) Lack of smart metering and billing. Users are billed on *as used* basis.
- 10) Lack of absolute control on data and computing assets custody
- 11) Potential to abuse the relative anonymity behind registration and usage models
- 12) Lack of detailed and fine grained resource monitoring mechanisms
- 13) Difficult to assess software licensing structure, especially in complex deployment scenarios where multiple licenses of different software are used in a single environment.

Service Model Specific Characteristics:

PaaS Cloud Specific:

- 1) Allows only provider supported programming languages, tools, APIs and components to build applications.
- 2) Can control deployed applications and possibly its hosting environment configurations.
- 3) No control of underlying infrastructure (network, servers, operating systems, or storage).

IaaS Cloud Specific: An IaaS cloud platform provides basic compute infrastructure as a VM plus some virtual storage and networking.

- 1) Allows resource utilization monitoring and reacting to events
- 2) Cloud user responsible for installing/managing all software on VM
- 3) Applications running in the VM are responsible for dealing with the reactions to above mentioned events. For instance, a configured reaction for a “CPU utilization threshold reached event” may be to add more instances of the VM. It is then expected that the architecture of the application(s) running on the VM allows to harness the newly added VM's capacity.
- 4) Limited control on networking components, e.g. host firewalls.

SaaS Cloud Specific:

- 1) Allows control of a limited set of user-specific application configuration settings.
- 2) No control of underlying infrastructure (network, servers, operating systems, storage, or individual application capabilities).

Deployment Model Specific Characteristics:

Public Cloud Specific:

- 1) Cloud service provider has the custody and control of applications, data and computing assets hosted on cloud.
- 2) Single point ownership of cloud infrastructure lies with the organization selling cloud services.
- 3) Often has homogeneous virtualization environment.
- 4) Allows limited configurations of cloud infrastructure.
- 5) Cloud infrastructure is made available to the general public for a fee.

Private Cloud Specific:

- 1) Often has a homogeneous virtualization environment.
- 2) Total ownership, control and custody of applications, data and computing assets.
- 3) Allows custom configurations of cloud infrastructure.
- 4) Operated solely for one organization.

Hybrid Cloud Specific:

- 1) A logical arrangement that combines two or more disparate clouds (private, community, or public).
- 2) Each constituent cloud remains a unique entity retaining its own characteristics.
- 3) Constituent clouds are linked via a technology (standardized or proprietary) that enables data and application portability.

Community Cloud Specific:

- 1) Member organizations have total ownership, control and custody of applications, data and computing assets.
- 2) Supports a specific community that has shared goals/concerns.
- 3) Distributed ownership of cloud infrastructure shared by several participating organizations.
- 4) Often has a homogeneous virtualization environment.

III. IMPACT OF PLATFORM'S CHARACTERISTICS ON QAS

The ability of any software system to achieve certain QAs is determined by:

- 1) Characteristics of the underlying platform on which the system is built
- 2) How the said characteristics have been harnessed (or mitigated) when designing architecture of the system.

The characteristics that we refer to above are the result of design decisions taken by architects of the platform in question. Architecture of a well designed software system typically implements some architectural pattern(s). Architectural patterns employ proven design tactics for addressing design concerns and achieving the desired levels of QAs [17], [18].

Therefore, by examining platform characteristics in light of existing body of architecture knowledge, their impact on various QAs can be easily found. For our study of the QA impact of various platform characteristics, we chose a subset of QAs listed in first part of the standard, ISO/IEC 9126-1. A partial list of QAs is:

Assets custody	Operability
Auditability	Performance isolation
Availability	Policies Compliance
Backup	Portability
Configurability	Privacy
Deployment	Reliability (MTBF)
Disaster recovery	Resource elasticity
Efficiency	Response time
Environmental impact	Scalability
Failure management	Security
Installability	Supportability
Interoperability	Tenant isolation
Maintainability	Testability
Modifiability	Throughput

In subsequent sections, we have determined the impact of platform characteristics identified in Section II on above set of QAs. We examined in detail the architecture patterns, design tactics and best practices knowledge from sources such as [17], [19]–[24] in order to determine the said impact on QAs. The information about impact of computing platform's characteristics on QAs is presented in the tabular (matrix) form. Along the rows are listed characteristics of the platforms, and along columns are indicated QAs.

Impact of a platform characteristic on a QA can be located at the intersection of respective row and column. A value of 1 for impact indicates that the characteristic in selected row has favourable impact on the QA in the column. Value -1 indicates an adverse impact on the QA, and the empty cell means that there is no impact.

A. Impact of Virtualization Based Platform Characteristics

Impact information for only a subset of all possible combinations between platform characteristics and QAs are described in detail here. The Table I, however, shows the impact information for all cases. It is easy to observe that as we move from parent to the subcategories of virtualization types such as VMM based, OS based and further down, the impact on QAs gets narrowed down to fewer QAs due to specialization.

1) *Ability to Take VM Snapshots:* Virtualization provides several capabilities to enable programmatic manipulation of VMs. It allows saving the state of a live or off-line VM to a file by taking a snapshot of the VM. This is similar to check-pointing of in-flight transactions in database systems. One can easily restore the VM to a prior good known state by restoring a snapshot on detecting a failure. This favourably impacts reliability, disaster recovery, backup and deployment etc. Table I shows impact on rest of the QAs.

2) *Abstraction of Hardware Resources as Software Entities:* The hardware resources such as CPU, memory, disk etc., are software entities in a virtualization based platform. For instance, CPU seen inside the VM is a representation of time-slices on underlying physical CPU cores. Design tactic of *abstraction* has been applied to present physical resources as software entities to the applications. This idea allows the VMs to be programmatically examined, controlled, saved and moved around over the network like regular files. Almost all

TABLE I
QA IMPACT FOR VIRTUALIZATION PLATFORMS

Characteristics ↓	Auditability	Availability	Backup	Configurability	Deployment	Disaster recovery	Efficiency	Environmental impact	Failure management	Instability	Interoperability	Maintainability	Modifiability	Operability	Policies Compliance	Portability	Privacy	Reliability (MTBF)	Response time	Scalability	Security	Supportability	Testability	Throughput	Resource elasticity	Assets custody	Tenant isolation	Performance isolation
Common across all platforms																												
Software abstraction of hardware	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1			1	-1	1		1	1	-1	1			-1
Limited hardware control capabilities for Vms							-1	-1									-1	-1				-1						
Programmatic self-serviced provisioning		1	1		1	1			1	1	1							1		1		1			1			
Multi-tenancy by logical partitioning of physical host					1	1	1	1	1	1	1						1	1	1	1	1	1	1	1	1		1	-1
VM check-pointing and snap shots		1	1		1	1	1	1	1	1		1	1			1		1		1	1	1	1	1	1	1		
VM migration (both live and offline)		1	1		1	1	1	1	1	1		1				1		1	1	1	1	1		1	1			
Abstraction of platform specific APIs and ABIs										1	1	1	1			1			-1						-1	1		
VMM Virtualization platform																												
Allows full guest OS control in the VM	1	1		1	1	1									1										1			
Uses block-level copy-on-write			-1																-1									
No restrictions on guest OS in VM				1																					1			
Programs in VMs use VMM provided emulation instead of the OSs normal system call interface							-1												-1					-1				-1
Bare-metal VMM platform																												
Thin and encapsulated h/w facing layer							1												1	1				1				
Hosted VMM platform																												
VM runs a regular process inside host OS	1	-1		1	1	1			1	1				1	1	1	-1		-1									-1
Can run unmodification guest OS in a VM										1						1												
OS Virtualization platform																												
Isolation under a shared OS instance only				-1			1		-1						1	1		-1	-1	1		-1	1	1	1			
Privileged tasks only under the host OS control	-1			-1	-1				-1	-1												1	-1					
Programs in VMs use the OSs normal system call interface													-1						1					1				
Tight integration with host OS												-1	-1				-1											
Guest OS (or kernel in some cases) in VM must be same as host.				-1									-1				-1											
Uses file-level copy-on-write mechanisms																				1								

the QAs, except some performance related QAs, that we listed are favourably impacted by this characteristic. For instance, backup, disaster recovery, operability etc. are favourably impacted. On the other hand, the additional layer of abstraction introduces performance penalties; as such response time and throughput are expected to be adversely impacted.

B. Impact of Cloud Based Platform Characteristics

Several variants of cloud based platforms exist, mainly differing based on service (i.e. IaaS, PaaS etc.) and deployment models (i.e. private, public etc.). They have many of the characteristics common, whereas some are specific to each variant. There are large number of combinations possible between platform characteristics and QAs. To highlight our analysis approach, we discuss the impact of only a subset of characteristics on QAs. Impact information for rest of the combinations is, however, presented in Table II.

1) *Limited Control of Underlying Platform*: The cloud users get only a limited control on underlying platform infrastructure in all service model based non-private variants of cloud. For instance, IaaS host machine's power management functions are typically not available in VMs. That is, a VM is not allowed to put the physical CPU to a low power state when idle. Similarly, on PaaS, only deployed applications and its hosting environment configurations can be controlled by applications. As such, this characteristic adversely impacts

adaptability, configurability and failure management. Most of the remaining QAs are not directly impacted and are as shown in Table II.

2) *Self-service Provisioning*: Cloud platforms provide programmatic APIs to allow automation of several common provisioning tasks. For instance, a VM with 4 CPUs, 8GB RAM, 250GB of disk can be created in seconds. This characteristic improves deployment, configurability, operability, backup etc. This is because achieving these QAs now does not require human/manual intervention. These tasks can now be performed via programmatic means by utilizing cloud APIs. We have determined the impact on other QAs by a similar analysis, and is shown in Table II.

IV. CONCLUSION

We have presented the detailed examination and analysis of various platforms, from the standpoint of guest application's architecture and design. Both virtualization and cloud platforms possess characteristics that impact the ability of guest applications to achieve certain QAs. For instance, the ability to take a snapshot of a running VM impacts the disaster recovery in a favourable manner. Similarly, the lack of physical custody of data and software assets by the users in case of cloud platforms impacts the security and privacy QAs adversely. We also observe that certain QAs such as resource elasticity, tenant isolation and performance isolation arise mainly in case of

TABLE II
QA IMPACT OF CLOUD PLATFORMS

Characteristics ↓	Auditiability	Availability	Backup	Configurability	Deployment	Disaster recovery	Efficiency	Environmental impact	Failure management	Installability	Interoperability	Maintainability	Modifiability	Operability	Policies Compliance	Portability	Privacy	Reliability (MTBF)	Response time	Scalability	Security	Supportability	Testability	Throughput	Resource elasticity	Assets custody	Tenant isolation	Performance isolation
Common across all cloud platforms																												
Relative anonymity behind subscription and usage																	-1				-1							
Programmatic provisioning of resources		1	1	1	1	1	1	1	1	1				1				1	1	1		1	1	1	1			
Allows self-service provisioning		1	1	1	1	1	1	1	1	1				1				1	1	1	-1	1	1	1	1			
Lack of absolute control on software/data assets custody	-1	-1							-1						-1		-1				-1							
Lack of standards for key services	-1										-1	-1	-1		-1	-1												
Measured service																												
Lack of smart metering and billing. Users billed on as used basis																									-1			
Lack of fine grained resource monitoring mechanisms	-1					-1			-1			-1			-1			-1			-1	-1						
Computing as a utility accessible over the network					1	1	1	1																		1		
Geographic location transparent to clients															-1		-1				-1							
Political/legal jurisdiction transparent to clients															-1		-1				-1							
Difficult to assess software licensing structure																												
PaaS platform																												
Only deployed applications and its hosting environment configurations can be controlled	-1			-1					-1																			
Only provider supported programming languages, tools, APIs etc. Allowed	-1										-1	1	-1			-1												
No control/responsibility of underlying infrastructure (hardware, OS)	-1			-1						1							-1				-1							
SaaS platform																												
Can control only a limited set of user-specific application configuration settings.	-1			-1									-1															
Provider specific service implementation																-1	-1											
IaaS platform																												
User responsible for installing/managing all software on VM										1				-1														
Allows resource utilization monitoring and reacting to events		1		1			1	1	1						1			1		1	1			1	1			
Applications in VM are responsible for handling above events									1							-1			1									
Limited control on networking components				-1																					-1			
Public Cloud platform																												
Infrastructure available to general public for fee																		-1			-1							
Cloud vendor has single point ownership of infrastructure																				-1								
Cloud provider retains custody and control of software/data assets																		-1								-1		
Private Cloud platform																												
Operated solely for one organization																										1		
Total ownership/custody of software/data assets	1					1												1			1					1		
Custom configurations of cloud infrastructure possible	1	1				1																						
Has a homogeneous virtualization environment						1	1				1				1										1			
Hybrid Cloud platform																												
Blending of two or more disparate clouds																		-1			-1					-1		
Community Cloud platform																												
Ownership of cloud infrastructure shared by participating organizations														-1														
Has a homogeneous virtualization environment						1	1				1				1											1		
Member organizations have ownership and custody of software/data assets																	1			1							1	

cloud platforms.

We believe that these knowledge artifacts presented here will help in performing the rational technical evaluation of various platforms.

REFERENCES

- [1] L. Youseff *et al.*, "Toward a unified ontology of cloud computing," in *Grid Computing Environments Workshop, 2008. GCE'08.* IEEE, 2008, pp. 1–10.
- [2] B. Rimal *et al.*, "Architectural requirements for cloud computing systems: An enterprise cloud approach," *Journal of Grid Computing*, pp. 1–24, 2011.
- [3] Oracle, "Consolidating applications with oracle solaris containers," <http://www.oracle.com/technetwork/server-storage/solaris/documentation/consolidating-apps-163572.pdf>, Oracle, May 2010, retrieved: August 2011.
- [4] Citrix / Xen, "Xen hypervisor - leading open source hypervisor for servers," <http://xen.org/products/xenhyp.html>, Citrix Systems Inc., retrieved: August 2011.
- [5] IBM Systems, "Virtualization (version 2 release 1)," <http://publib.boulder.ibm.com/infocenter/eserver/v1r2/topic/eicay/eicay.pdf>, IBM, retrieved: August 2011.
- [6] A. Chobotaro *et al.*, "New client virtualization usage models using intel[®] virtualization technology," *Intel[®] Technology Journal*, vol. 10, no. 3, August 2006.
- [7] M. Rosenblum and T. Garfinkel, "Virtual machine monitors: current technology and future trends," *Computer*, vol. 38, no. 5, pp. 39 – 47, May 2005.
- [8] M. Bolte *et al.*, "Non-intrusive virtualization management using libvirt," in *Design, Automation Test in Europe Conference Exhibition (DATE), 2010*, march 2010, pp. 574 –579.
- [9] Amazon, "Amazon web services documentation," <https://aws.amazon.com/documentation/>, Amazon Inc., August 2011.
- [10] D. Nurmi *et al.*, "The eucalyptus open-source cloud-computing system," in *Proceedings of the 2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid*, ser. CCGRID '09. Washington, DC, USA: IEEE Computer Society, 2009, pp. 124–131.
- [11] K. Keahey *et al.*, "Sky computing," *Internet Computing, IEEE*, vol. 13, no. 5, pp. 43 –51, Sept 2009.
- [12] OpenNebula, "Opennebula 2.2 documentation," <http://www.opennebula.org/documentation:documentation>, OpenNebula Project, retrieved: August 2011.
- [13] C. . Xen, "Xen cloud platform - open source software to build private and public clouds," <http://xen.org/products/cloudxen.html>, Citrix Systems Inc., retrieved: August 2011.
- [14] Microsoft, "Develop applications - code and configure applications to run in windows azure," <http://www.microsoft.com/windowsazure/learn/develop/>, Microsoft Inc., retrieved: August 2011.
- [15] Google, "Google app engine developer's guide," <http://code.google.com/appengine/docs/>, RaceLab, UCSB, retrieved: August 2011.
- [16] N. Chohan *et al.*, "Appscale: Scalable and open appengine application development and deployment," *Cloud Computing*, pp. 57–70, 2010.
- [17] L. Bass *et al.*, *Software architecture in practice*. Addison-Wesley Longman Publishing Co., Inc., 2003.
- [18] N. Harrison and P. Aygeriou, "How do architecture patterns and tactics interact? a model and annotation," *Journal of Systems and Software*, vol. 83, no. 10, pp. 1735–1758, 2010.
- [19] E. Gamma *et al.*, *Design patterns*. Addison-Wesley Reading, MA, 2002, vol. 1.
- [20] D. Schmidt *et al.*, *Pattern-Oriented Software Architecture: Patterns for Concurrent and Networked Objects, Volume 2*. Wiley, 2000.
- [21] M. Kircher and P. Jain, *Pattern-Oriented Software Architecture Volume 3: Patterns for Resource Management*. Wiley, 2004.
- [22] F. Buschmann *et al.*, *Pattern-Oriented Software Architecture Volume 5*. Wiley-India, 2007, vol. 5.
- [23] D. Alur, J. Crupi, and D. Malks, *Core J2EE patterns: best practices and design strategies*. Prentice Hall PTR, 2003.
- [24] G. Abowd, L. Bass *et al.*, "Recommended best industrial practice for software architecture evaluation." DTIC Document, Tech. Rep., 1997.